A Comparison of Different Computer Vision Methods and Algorithms for the Classification of Aquatic Macroinvertebrates

[*]A Computer Vision project for the Department of Engineering at Aarhus University

Théo Morales
MSc. Student of Computer Engineering
Faculty of Science and Technology, Department of Engineering
Aarhus University, Denmark
theo.martin.morales@post.au.dk

***Abstract*—Measuring water quality in natural environments can be a difficult problem to tackle. One of the ways of assessing the quality of fresh water in such environment is to observe and analyse the different aquatic macroinvertebrates that live in it. Using modern Computer Vision methods to do so is inherently more efficient, in terms of time and cost, than relying on the sole human expertise. Developing proper techniques in order to achieve similar, if not better, results than manual observation and analysis is the ultimate goal of this research. In this paper, a subset of the original dataset is used to conduct independent research on different image classification algorithms, and to compare several common ones with the state of the art Convolutional Neural Networks.**

***Index Terms*—CNN, Deep Neural Network, image classification, computer vision, machine learning**

I. Introduction

Ensuring the quality of water sources is an important task for the good of a human population, as well as the one of its surrounding ecosystem. Water can be infected with all sorts of bacteria, but can also contain a significant amount of micro-organisms that wouldn't be suited for human consumption, but nevertheless being part of a natural aquatic ecosystem that could be seen as a gage of quality and sanity.

However, measuring such concept isn't as trivial as what is commonly done with regular metrics, and it requires more reasoning and analysing than most standard measures in the scientific domain do. This study focuses on the analysis of known aquatic macroinvertebrates, which are part of aquatic ecosystems in water sources, and their automated classification using a machine learning approach. The end goal is to provide a reliable method for this task, that would ultimately surpass the human expertise in the field. In this paper, a set of well known and commonly used machine learning and computer vision methods are presented, and their results are compared and discussed after application on the dataset; the famous state-of-the-art Convolutional Neural Network will be used as a reference point in the benchmark.

II. Dataset

A. Dataset presentation

The provided dataset contains a subset of the whole dataset from the original research paper that this project leans on. It contains **5830** *Training* samples, **2298** *Validation* samples, and **3560** *Test* samples. These sets are given in two forms:

- the original images in color with a **28x28** pixel dimension, in the JPEG format
- the abstract representations of the images as a set of 4096-dimensional vectors (available inn *CSV*, *Matlab* and *text* formats), obtained after feeding the base pictures to a CNN, pre-trained on the training dataset

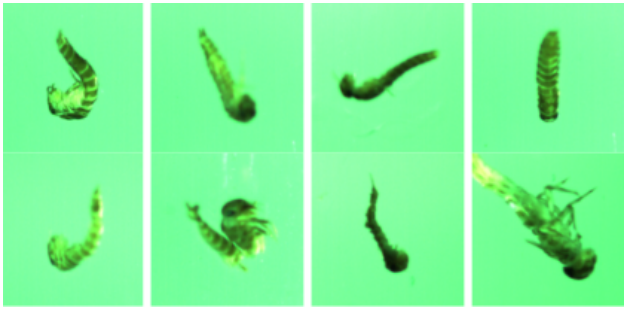

Fig. 1. Four samples from Baetis niger (top) and Ameletus inopinatus (bottom) classes.

It goes without saying that each set also comes with its corresponding labels definition (available in *CSV*, *Matlab* and *text* formats), except for the *Test* dataset which labels, of course, need to be "guessed" by the classifier.

B. Dataset pre-processing

One important thing to mention is that the given JPEG pictures are already cropped to focus on the target object, thus minimizing the need for data pre-processing in some cases where not much needs to be done. For example, with computer vision methods such as *SIFT* or *HOG* features classification,

a minimal amount of data preparation is necessary for the algorithms to be effective: vector normalization is often just enough. In this study, the *Feature Scaling* normalization technique is used, in order to bring all the dataset's values into the range [0,1]:

$$x_{new} = \frac{x_i - \min(x)}{\max(x) - \min(x)} \tag{1}$$

With x_i being the current value of the sample (a vector in this case), $\min(x)$ being the minimum value of the current dataset, and $\max(x)$ being the maximum value of the same dataset. This technique is useful when the classifier is computing a lot of multiplications, since it greatly reduces the difference between the two products when the multiplied values are very close to each other, thus restraining the scatter of the samples.

Data standardization can also be used in order to improve the classification results, by reducing the mean and unit variance to zero, using the following equation:

$$x_{new} = \frac{x_i - \mu}{\sigma} \tag{2}$$

Where μ is the mean of the dataset, and σ is the variance of the same dataset.

Concerning the Convolutional Neural Networks approach, a few more things have to be taken into consideration. In this project, a pre-trained CNN is used for fine-tuning over this dataset, meaning that the weights of the convolution layers are unchanged. This implies that the images fed into the network must be the same size as the ones it has been trained with. Without going into further details (more will be explained in the appropriate section), the JPEG images for this dataset had to be scaled up to match the network's input layer (299x299 pixels), and some distortion is applied to the *Training* images. Other than that, each image is normalized as previously explained.

C. Dataset augmentation

As stated above, *Dataset Augmentation* is a process that is only applied to images used for a Convolutional Neural Network training phase, because it is the classifier that takes the most advantage of the three color channels of an image. It is only useful to augment the *Training* sample images, since those are the only ones that will help the classifier to learn. The *Python* script provided by *Tensorflow* for *Inception Resnet v2* tales care of distorting images in order to augment the data set, to ultimately make the network invariant to aspects of the image that do not effect the label. Since the aspect ratio is not respected, and since the resizing method changes depending on the running thread number, the resizing operation may distort the images in a first place. Secondly, each image is randomly flipped horizontally. Finally, the colors are randomly distorted in four different ways.

III. Methods and algorithms

Classifying images of such similar shapes can be a difficult task to solve as a machine learning problem, and that is why several algorithms and methods must be applied on the same dataset in order to have a certain order of comparison to be able to conclude useful deductions. Finding the right method to properly classify the dataset is the ultimate goal, but before simply going for the state-of-the-art CNN, it is best to study the well known methods and understand why one performs better than an other, in an attempt to apply the best possible approach.

The following algorithms have been applied and compared against each other:

- **Multi Layer Perceptron Neural Network**
- **Pre-Trained Convolutional Neural Network**
- **Support Vector Machine with Bag Of Features**
- **Nearest Neighbour with Bag Of Features**

Those algorithms were implemented in *Python 3*, with the help of the great libraries *Tensorflow* and *OpenCV*. In the following section, they will be presented and explained thoroughly, after what their results will be compared on the same basis. *Note that the given accuracy corresponds to the validation accuracy, and that it may vary if measured on the test dataset.*

A. Multi Layer Perceptron Neural Network

Famously known for being able to optimize a large range of cost functions, the MLP Neural Network does not need to be presented. The thing that makes it a very good candidate against the convolutional type is its fully-connected architecture, that is part of the actual CNN architecture as being the last "step". Since the feature vectors provided in the dataset were produced by a CNN, it should be intuitive to think that feeding those into an independent MLP would produce rather similar results, and that is why it can be assumed that both scores should be close to each other.

The MLP has been trained from scratch using different architecures for the hidden layers, and with two different stochastic optimizers: *Adam* and *Stochastic Gradient Descent*, both with similar learning rates. As for the activation function, $ReLU$ (for Rectified Linear Units) is used for the hidden layers, which, as research has shown [1], results in much faster training for large network and is more similar to an actual neuron in the human brain.

$$f(x) = \max(x, 0) \tag{3}$$

However for the output layer, the $Softmax$ activation function was used, since it is more suited to deal with classification problems [2]. It works similarly to $Sigmoid$, in a way that it will squash the inputs to be between 0 and 1, but it is more equivalent to a categorical probability distribution:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{k} e^{z_k}} \tag{4}$$

Different results can be observed after tinkering with the hyperparameters, before obtaining what is to be considered the optimal configuration for the last row of the table:

TABLE I
MULTI LAYER PERCEPTRON:
HYPERPARAMETERS AND CONFIGURATIONS

Hyperparameters			**Final**
Hidden Neurons	***Learn. Rate***	***Batch Size***	***Accuracy***
2048:1024	0.0018	50	72%

B. Pre-Trained Convolutional Neural Network

As previously explained, using a pre-trained Convolutional Neural Network has the big advantage of providing well-performing convolution layers, which play the biggest part in image classification for such a type of neural network. The chosen CNN is *Inception Resnet V2*, that is part of *Tensorflow-slim*'s research architectures and one of the top performing CNNs out there, according to *Google*'s research [3].

A checkpoint of that network, containing all the weights and biases, was downloaded and used to transfer the learning of the model, by keeping the convolution layers's weights but dropping the fully-connected layers's ones, in order to fine-tune the network with the macroinvertebrates dataset. This was done fairly easily thanks to the good usability of the framework, and as stated above, a pre-processing script is provided for the images to be fitted into the network's input configuration.

This method is allegedly the best in the field, and if used well, can yield impressive results of accuracy above 95%. However, since the computation of such network (especially on a large model like *Inception*) is very slow and requires a recent GPU with CUDA capabilities, it has been hard to obtain decent results that would support the theory.

C. Support Vector Machine with Bag Of Features

The vectors provided in each partition of the dataset were produced by a CNN trained on thousands of classes, which means the convolution filters are able to recognize all sorts of shapes very well. Those feature or shape vectors can be directly classified and it would be reliable, however it is interesting to see if a nearly good result can be achieved with engineered features: features that are found in images using efficient algorithms.

A *Support Vector Machine* has been trained on the features obtained by the CNN, and another one by SIFT features computed from the original images, using the Bag Of Features (or Bag Of Words) technique.

a) CNN Features with SVM: Firstly, the given vectors were used to train an SVM using the kernel trick. The comparison between a linear SVM and a kernel SVM (using the *rbf* kernel) can be seen in the following table: endtable

Without surprise, the kernel SVM performs better, and normalizing the data helped improve the final accuracy a little.

b) SIFT Features with SVM: Secondly, features have been extracted "manually" using the *Scale-Invariant Feature Transform* (SIFT) computer vision algorithm, with the *OpenCV* library.a This algorithm can be resumed in 6 steps:

1) Image scale-space generation using *Gaussian Blur*
2) *Difference Of Gaussians* (DOG) computation
3) DOG extremas location and interpolation (to find correct subpixels)
4) Interest point suppression by threshold
5) Gradient orientation extraction from interet point neighbourhood
6) Invarient feature descriptor generation

After obtaining the SIFT features, the Bag Of Features technique is used to generate histograms of a consistent size, by clustering the feature descriptors using *K-Means*. Those histograms can then be classified, using an SVM in this current example.

D. Nearest Neighbour with Bag Of Features

The *Nearest Neighbour* algorithm is very simple and simply consists in comparing each testing vector (here the histogram obtained after BoF) with every training vector, using the euclidian distance defined as:

$$d_E(y,x) = ||y-x||_2 = \sqrt{(y-x)^\intercal (y-x)} \tag{5}$$

The shortest distance is then used to attribute the sample the label of the winning candidate.

a) CNN Features with Nearest Neighbour: Using the same process: Normalization also helps improving the accuracy.

b) SIFT Features with Nearest Neighbour: For the *Nearest Neighbour* algorithm, the process is very similar: apply Bag Of Features on the SIFT features, then classify them with the algorithm.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," http://www.cs.toronto.edu/

[2] Ji Yang, "ReLU and Softmax Activation Functions," https://github.com/Kulbear/deep-learning-nano-foundation/wiki/ReLU-and-Softmax-Activation-Functions, 2017

[3] Tensorflow, "TensorFlow-Slim image classification model library," https://github.com/tensorflow/models/tree/master/research/slim